US005542053A

United States Patent [19]

Bland et al.

[11] **Patent Number:** 5,542,053

[45] **Date of Patent:** Jul. 30, 1996

[54] **BRIDGE INTERFACE BETWEEN TWO BUSES OF A COMPUTER SYSTEM WITH A DIRECT MEMORY ACCESS CONTROLLER PROGRAMMED BY A SCATTER/GATHER PROGRAMMER**

[75] Inventors: **Patrick M. Bland**, Austin, Tex.; **Daniel R. Cronin, III**, Lake Worth, Fla.; **Richard G. Hofmann**, Cary, N.C.; **Dennis Moeller; Lance M. Venarchick**, both of Boca Raton, Fla.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **350,184**

[22] Filed: **Nov. 30, 1994**

[51] **Int. Cl.^{6}** **G06F 13/00**
[52] **U.S. Cl.** **395/309**; 395/280; 395/311
[58] **Field of Search** 395/280, 309, 395/311

[57] **ABSTRACT**

A bridge interface for a computer system having an industry standard architecture (ISA) bus and a peripheral controller interconnect (PCI) bus is coupled between the ISA and PCI buses. The bridge has a direct memory access (DMA) control circuit programmable by programming signals to perform a DMA transfer, and a scatter/gather unit coupled between the ISA bus and the DMA control circuit. The scatter/gather unit selectively provides the programming signals to the DMA control circuit directly or causes the programming signals to be provided over the ISA bus. Providing the programming signals to the DMA control circuit directly from the programming controller of the scatter/gather unit takes advantage of the location of both the DMA control circuit and the scatter/gather unit on the bridge chip. The program controller is able to program the DMA control circuit with I/O write cycles that take only one 33 MHz clock, instead of a plurality of 8 MHz clocks if the DMA control circuit is programmed with signals sent over the ISA bus. This provides an order of magnitude reduction is the time required to program the DMA control circuit.

14 Claims, 10 Drawing Sheets
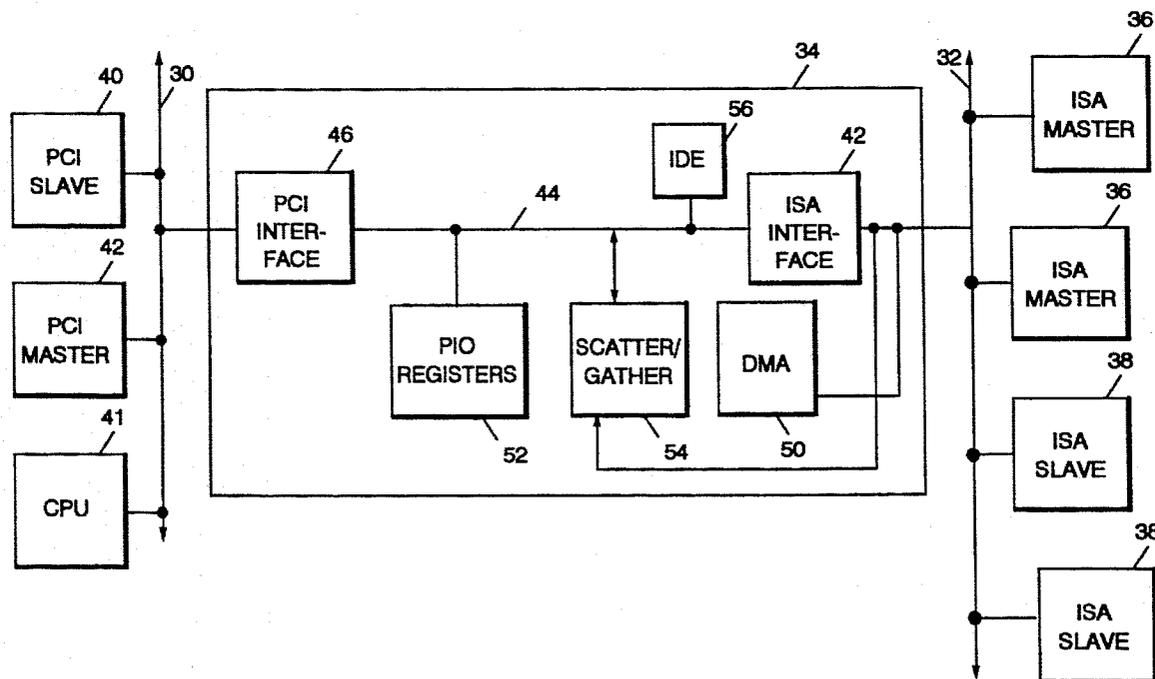
FIG. 1
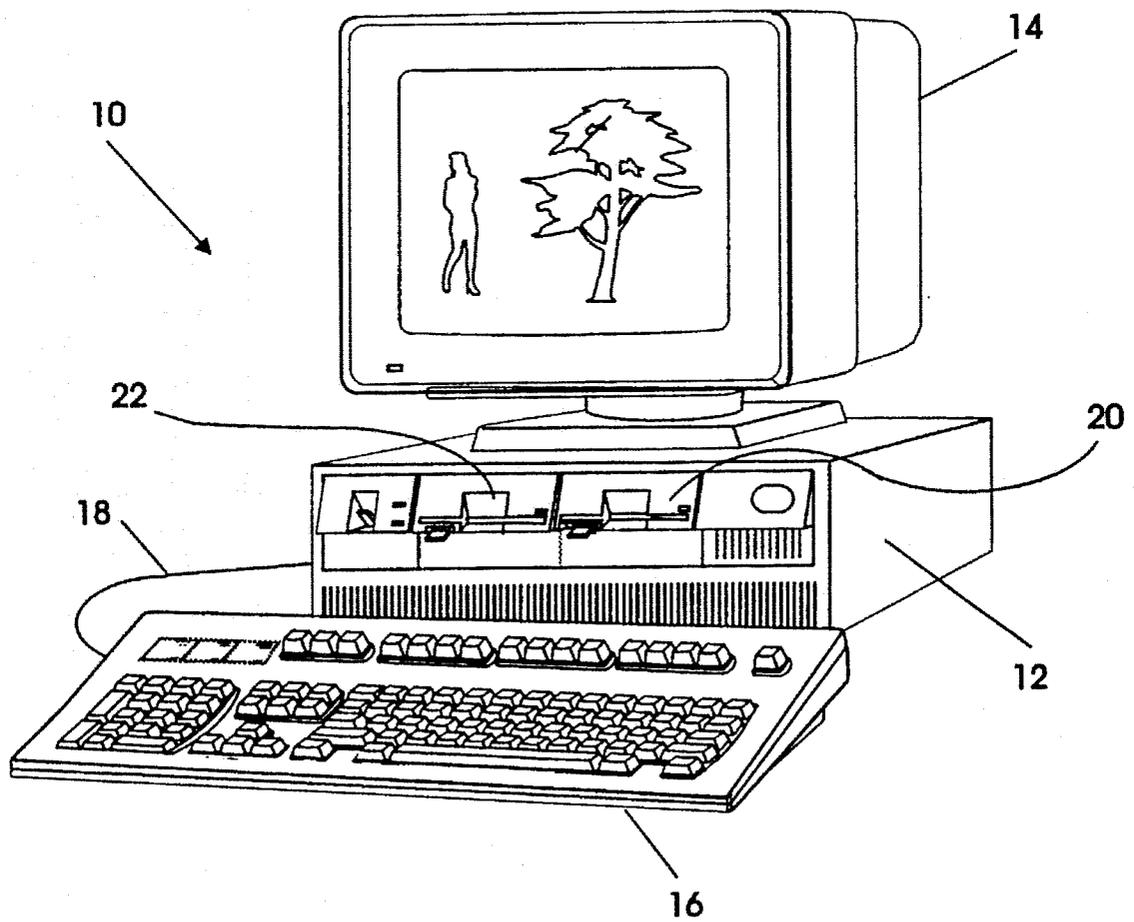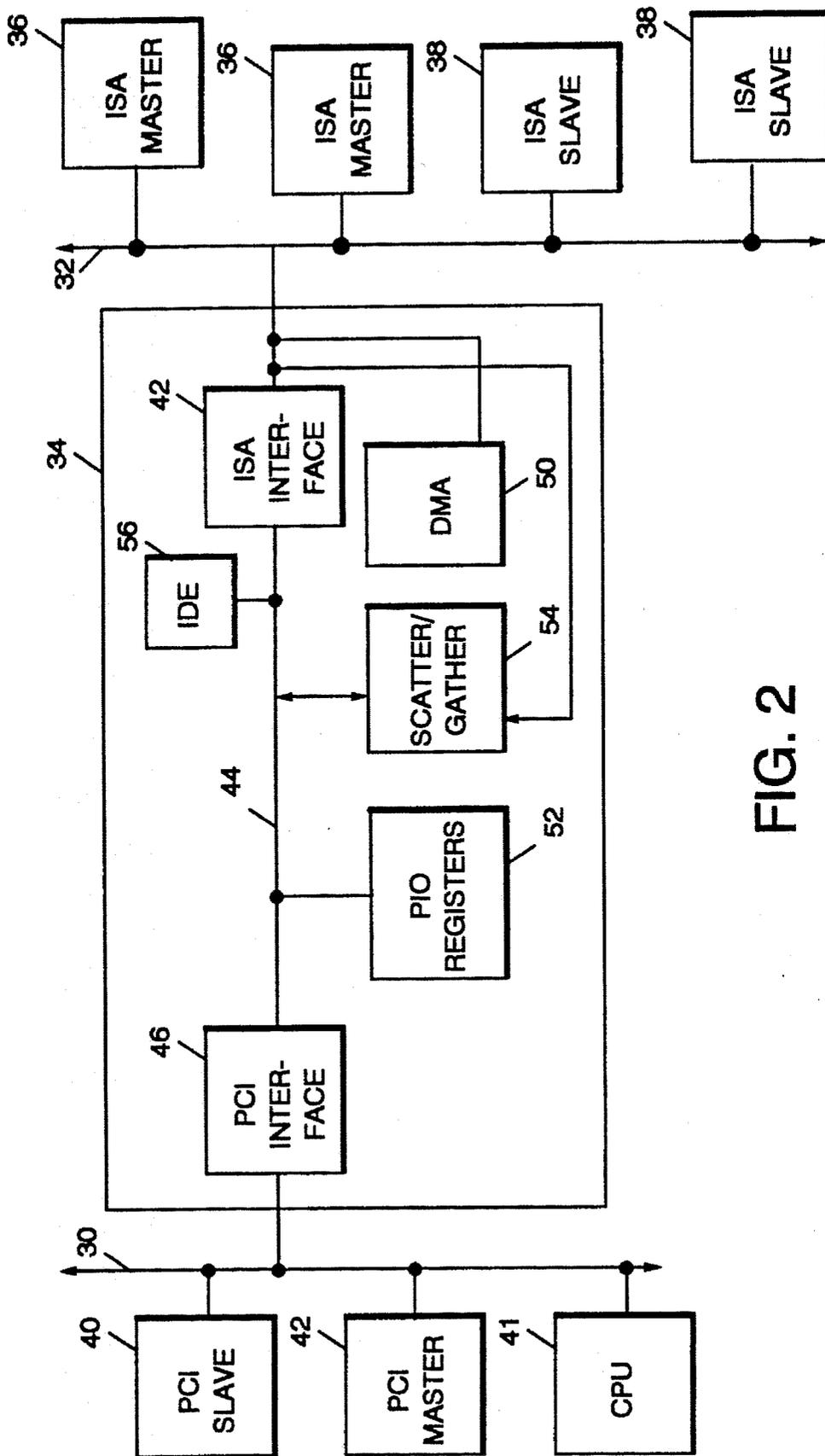
FIG. 2
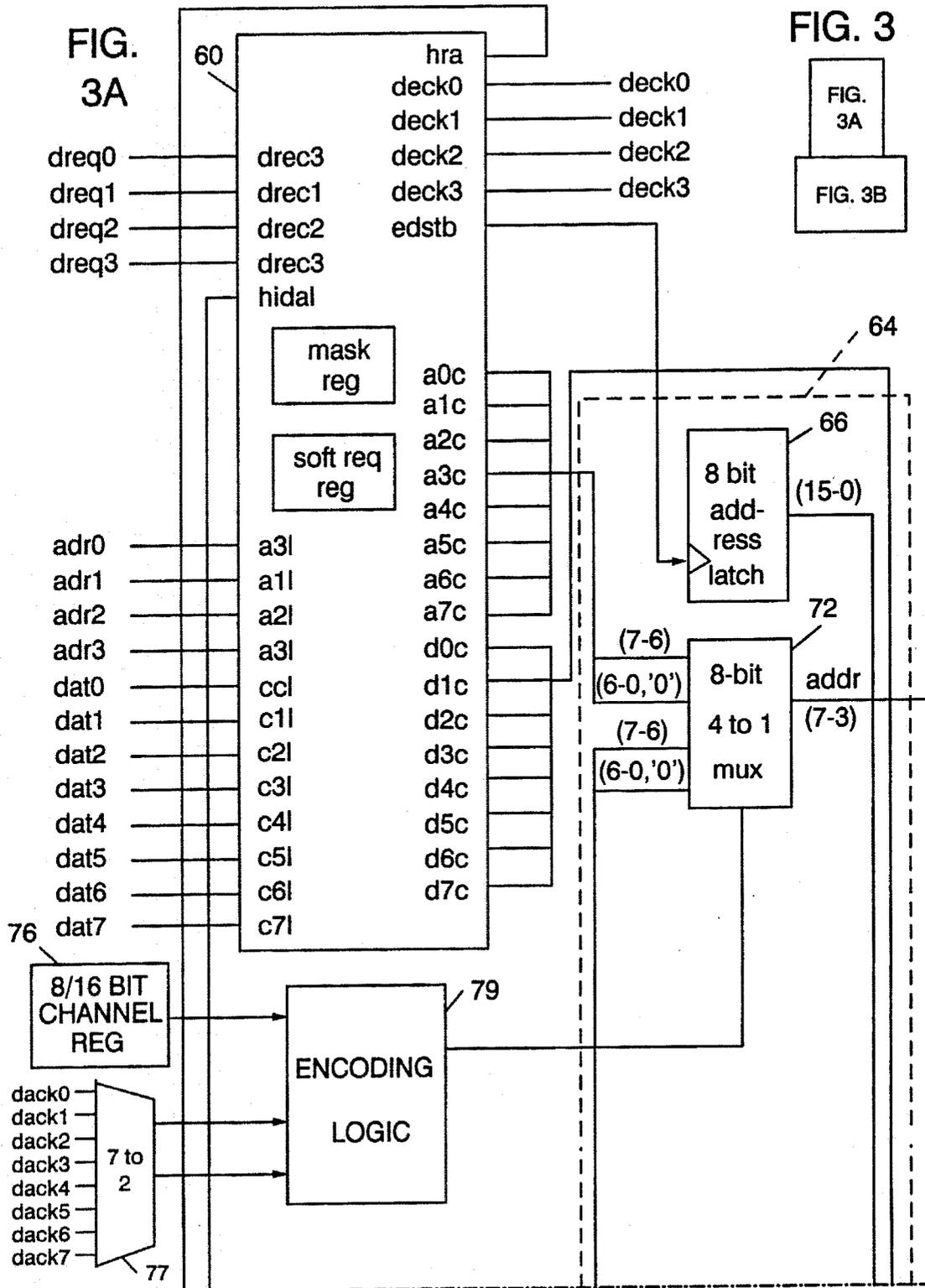
FIG.
3A
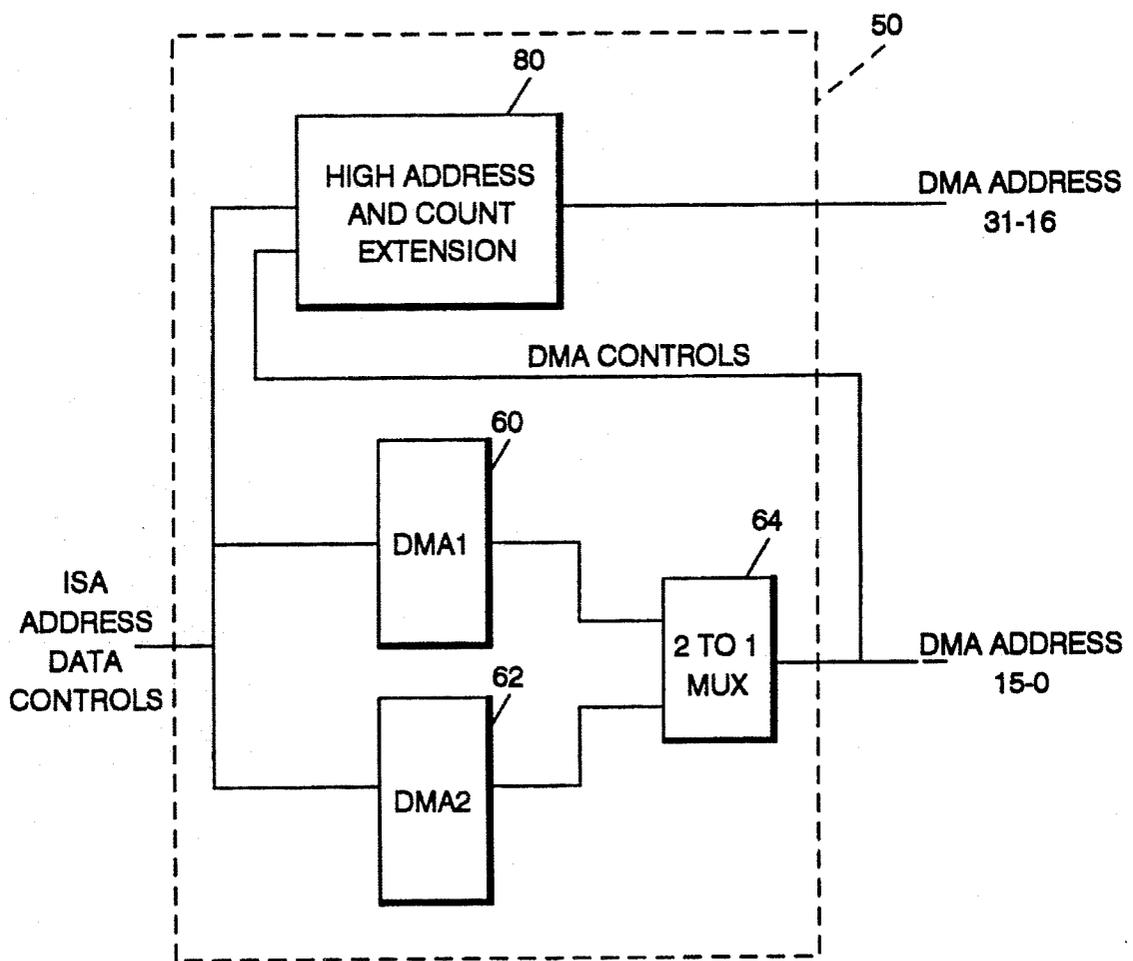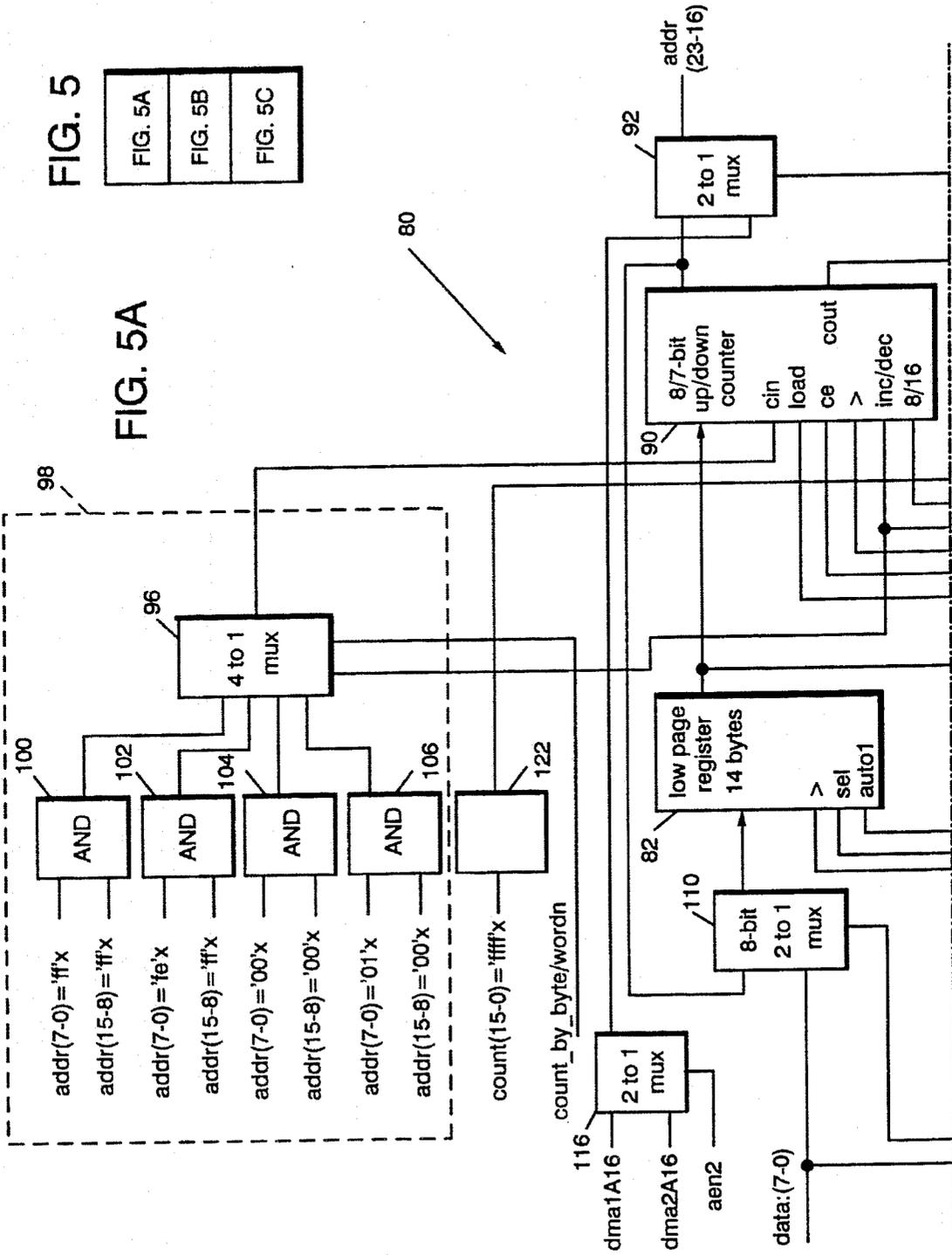
FIG. 3

FIG.
3A

FIG. 3B

FIG. 3B

FIG. 4

FIG. 5

FIG. 5A
FIG. 5B
FIG. 5C

FIG. 5A

FIG. 5B

FIG. 5C

FIG. 6

FIG. 7

CLK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

IOW#

ADDR	byteptr	current_address	lowpage	hi_page	byteptr	current_count	highcnt

DATA	chn1	addr0	addr1	addr2	addr3	chn1	cnt0	cnt0	cnt2

BRIDGE INTERFACE BETWEEN TWO BUSES OF A COMPUTER SYSTEM WITH A DIRECT MEMORY ACCESS CONTROLLER PROGRAMMED BY A SCATTER/GATHER PROGRAMMER

BACKGROUND OF THE INVENTION

1. Field of the Invention

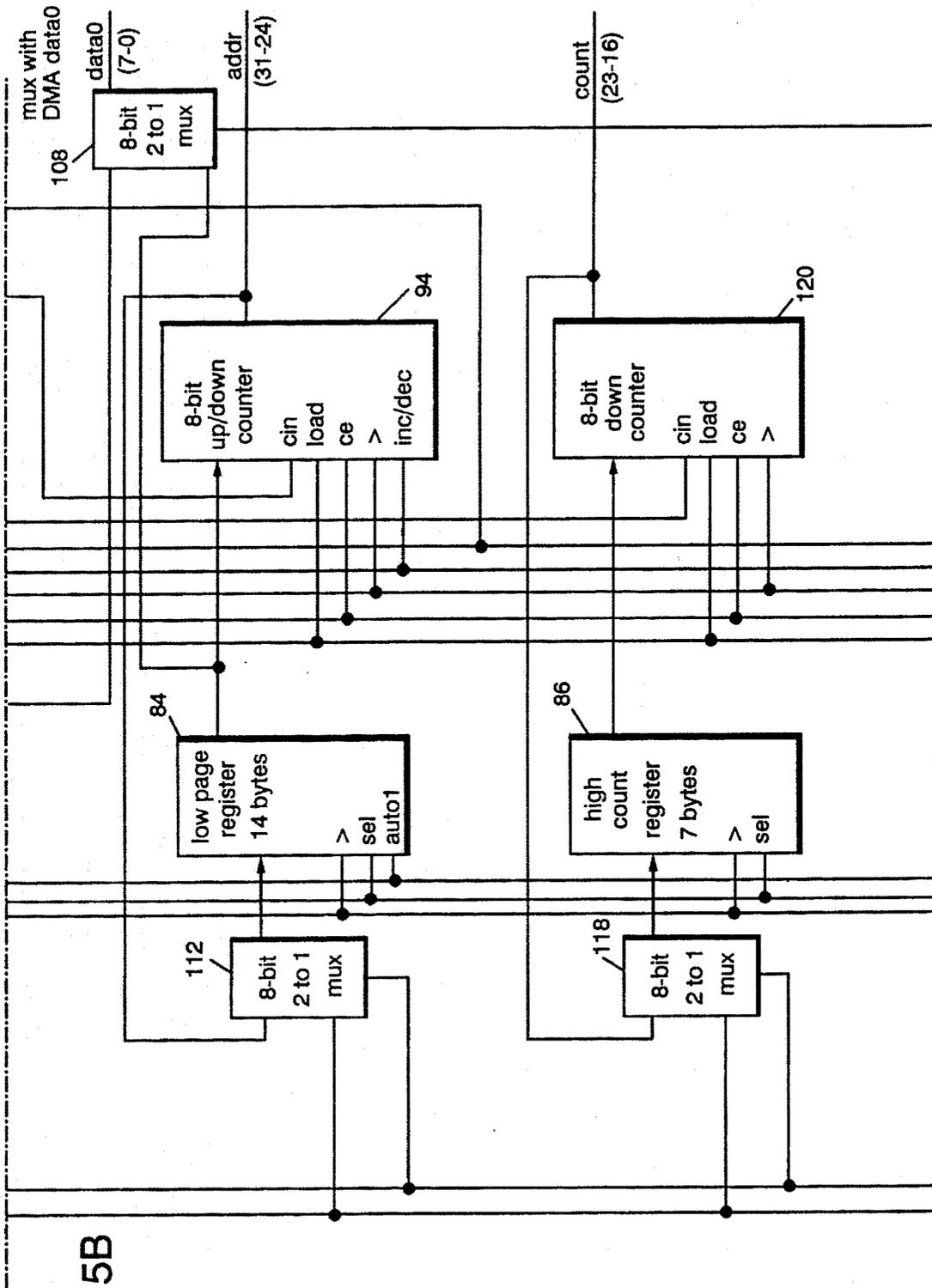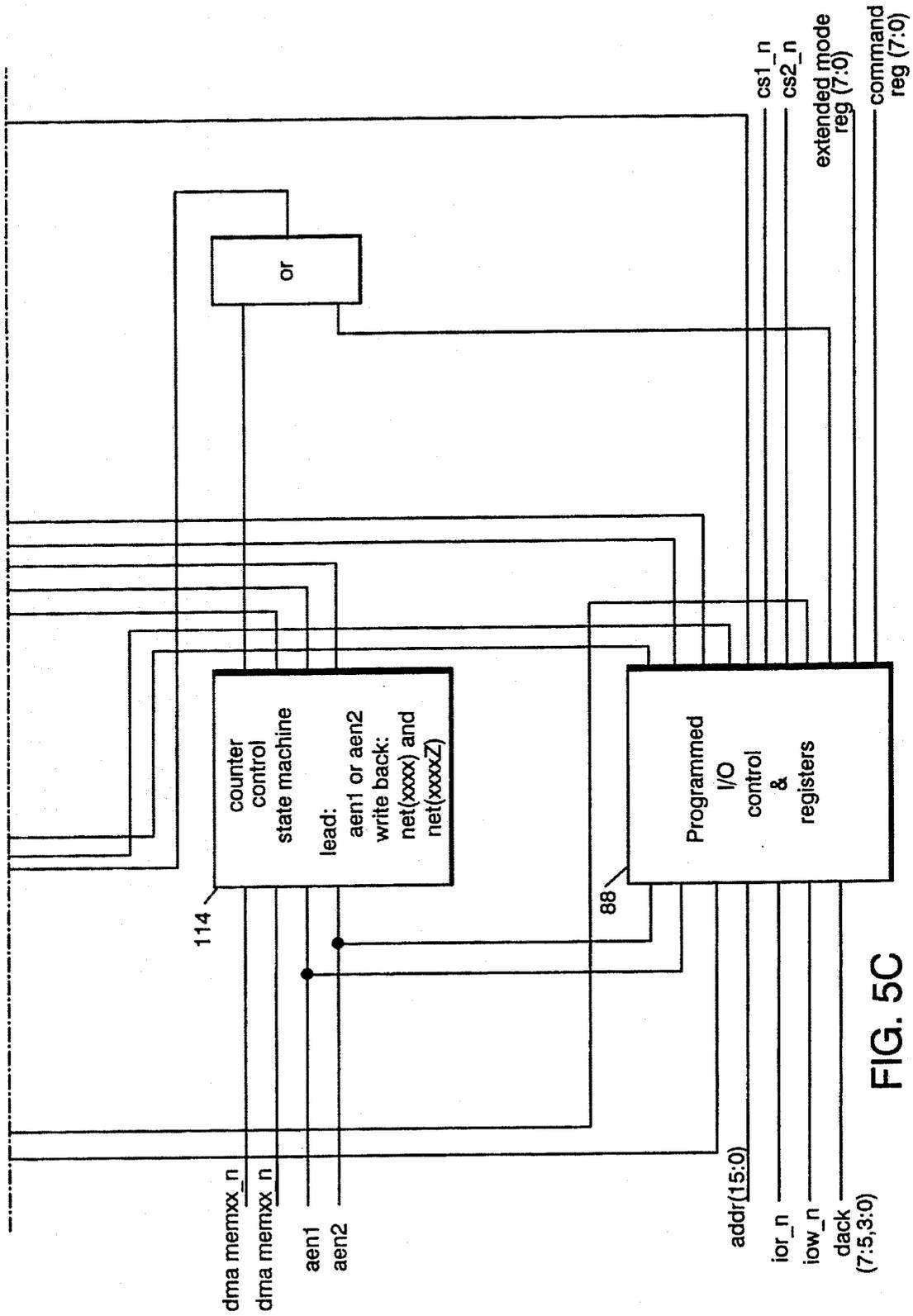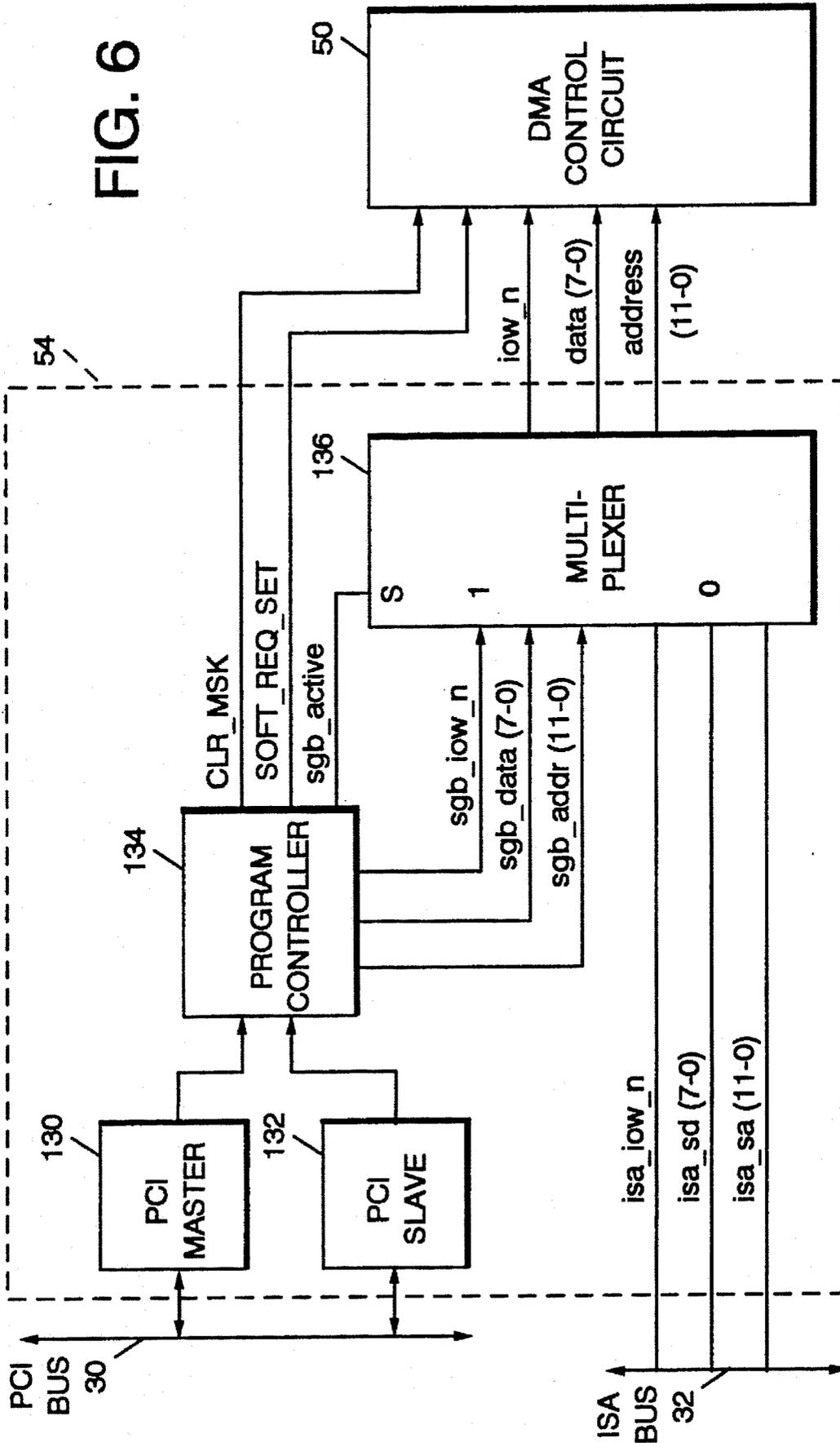The present invention relates to the field of digital computer systems, and more particularly, to the programming of a direct memory access controller located on a bridge between two buses by a scatter/gather programmer.

2. Description of Related Art

In computer systems, electronic chips and other components are connected with one another by buses. A variety of components can be connected to the bus providing inter-communication between all of the devices that are connected to the bus. One type of bus which has gained wide industry acceptance is the industry standard architecture (ISA) bus. The ISA bus has twenty-four (24) memory address lines which therefore provides support for up to sixteen (16) megabytes of memory. The wide acceptance of the ISA bus has resulted in a very large percentage of devices being designed for use on the ISA bus. However, higher-speed input/output devices commonly used in computer systems require faster buses. A solution to the general problem of sending and receiving data from the processor to any high-speed input device is a local bus. Unlike the ISA bus, which operates relatively slowly with limited bandwidth, a local bus communicates at system speed and carries data in 32-bit blocks. Local bus machines remove from the main system bus those interfaces that need quick response, such as memory, display, and disk drives. One such local bus that is gaining acceptance in the industry is the peripheral component interconnect (PCI) bus. The PCI bus can be a 32 or 64-bit pathway for high-speed data transfer. Essentially, the PCI bus is a parallel data path provided in addition to an ISA bus. The system processor and memory can be attached directly to the PCI bus, for example. Other devices such as graphic display adapters, disk controllers, etc. can also attach directly or indirectly (e.g., through a host bridge) to the PCI bus.

A bridge chip is provided between the PCI bus and the ISA bus in order to provide communication between devices on the two buses. The bridge chip essentially translates the ISA bus cycles to PCI bus cycles, and vice versa.

Many of the devices attached to the PCI bus and the ISA bus are master devices that can conduct processing independently of the bus or other devices. Certain devices coupled to the buses are considered to be slaves or targets that accept commands and respond to requests of a master. The PCI bus has an addressing capability of 32 bits to provide for 4 gigabytes of memory access. A master on the ISA bus can access a memory location in the memory on the PCI bus, although normally only within the lowest 16 megabyte region due to the 24-bit addressing of the ISA bus masters.

In a digital computer, a microprocessor operates on data stored in a main memory. Since there are practical size limitations on the main memory, bulk memory storage devices are provided in addition to and separately from the main memory. When the microprocessor wants to make use of data stored in bulk storage, for example, a hard disk, the data is moved from the hard disk into the main memory. This movement of blocks of memory inside the computer is a

very time consuming process and would severely hamper the performance of the computer system if the microprocessor were to control the memory transfers itself.

In order to relieve the microprocessor from the chore of controlling the movement of blocks of memory inside the computer, a direct memory access (DMA) controller is normally used. The DMA controller receives information from the microprocessor as to the base location from where bytes are to be moved, the address to where these bytes should go, and the number of bytes to move. Once it has been programmed by the microprocessor, the DMA controller oversees the transfer of the memory data within the computer system. Normally, DMA operations are used to move data between input/output (I/O) devices and memory.

The programming of a DMA controller by a central processing unit (CPU) requires the intervention of the processor for a relatively long time, since program input/output (I/O) cycles are run with standard ISA cycles. The processor cannot run other programs when it is programming the DMA controller. To improve system efficiency, scatter/gather units have been developed which take over from the CPU the programming of the DMA controller.

A scatter/gather unit essentially fetches descriptors, which are small blocks of memory, and programs these descriptors into the DMA controller. A descriptor includes a DMA address, a transfer count, and an end of link bit. The DMA address indicates the address where the data will be sent to or retrieved from; the transfer count indicates how large a block of data will be transferred; and the end of link bit indicates whether or not chaining is being used. When chaining is used, so that the end of link bit is cleared, the scatter/gather unit recognizes that there is another descriptor with a subsequent eight bytes (contiguous addressing), which contains another scatter/gather transfer.

In operation, therefore, the CPU will program a scatter/gather descriptor table in memory, which requires much less time than writing to slow I/O devices. The CPU then sends an instruction to the DMA controller to instruct it to prepare to perform a transfer, and an instruction to the scatter/gather unit to begin the transfer. The CPU's role is complete at this point and the scatter/gather unit retrieves the descriptor information from memory and programs the DMA controller with the information. The DMA controller is activated and operates as usual. The DMA controller is not aware that it was programmed by a scatter/gather unit and not the CPU with slow I/O accesses.

The "scatter" operation is a DMA transfer from I/O to memory, so that data is being read from a peripheral I/O device or a floppy disk, etc., and is scattered into memory. In a "gather" operation, the DMA controller is programmed to read memory and write to I/O by gathering different non-contiguous segments of memory and then writing them to I/O in a continuous manner.

A known bridge chip used to bridge an ISA bus and a PCI bus that contains a scatter/gather unit is available from Intel Corporation. When programming the DMA controller that is also located on the bridge chip, the scatter/gather unit runs ISA bus cycles on the ISA bus. These bus cycles are I/O cycles that are run on a bus that operates at 8 MHz. In comparison to the 33 MHz speed of the PCI bus, the 8 MHz bus is extremely slow, so that the programming of the DMA controller by the scatter/gather unit in the known device is also extremely slow.

SUMMARY OF THE INVENTION

There is a need for a system that has a bridge chip with a scatter/gather unit that programs a DMA controller at a

higher speed than the 8 MHz speed of an ISA bus that is coupled to the bridge chip.

This and other needs are met by the present invention which provides a bridge interface between buses of a computer system having a first bus and a second bus, the bridge comprising a direct memory access (DMA) control circuit programmable by programming signals to perform a DMA transfer; and a scatter/gather unit coupled between the first bus and the DMA control circuit. The scatter/gather unit selectively provides the programming signals to the DMA control circuit directly or causes the programming signals to be provided over the first bus.

The above stated needs are also met by another embodiment of the present invention which provides a computer system comprising a first bus and a second bus, and a bridge coupled between the first and second buses. The bridge has a direct memory access (DMA) control circuit programmable by programming signals to perform a DMA transfer, and a scatter/gather unit coupled between the first bus and the DMA control circuit. The scatter/gather unit selectively provides the programming signals to the DMA control circuit directly or causes the programming signals to be provided over the first bus.

By providing the programming signals to the DMA control circuit directly from the programming controller of the scatter/gather unit, the present invention takes advantage of the location of both the DMA control circuit and the scatter/ gather unit on the bridge chip. The program controller does not need to send the programming signals over the first bus, which may operate with a lower clock rate (such as 8 MHz) than the clock rate of the bridge chip (33 MHz). The direct programming of the DMA control circuit by the scatter/ gather unit is performed with a plurality of I/O writes that are each performed in one 33 Mhz clock. By contrast, each of the plurality of I/O writes performed over the first bus, such as an ISA bus, requires a plurality of 8 MHz clocks. The present invention therefore provides approximately an order of magnitude reduction in the amount of time required to run I/O program cycles from the scatter/gather unit to the DMA control circuit compared to the prior art. The present invention allows the slower programming, however, in addition to the fast programming, to provide backward compatibility.

The foregoing and other objects, features, aspects and advantages of the present invention will become more apparent from the following detailed description of the present invention when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a perspective view of a computer system of the present invention.

FIG. 2 is a block diagram of the computer system of FIG. 1 constructed in accordance with an embodiment of the present invention.

FIG. 3 is a block diagram of an exemplary embodiment of a DMA control circuit constructed in accordance with an embodiment of the present invention.

FIG. 4 is a basic block diagram of a DMA control circuit constructed in accordance with an embodiment of the present invention having high address extension and high count extension logic.

FIG. 5 is a block diagram of an exemplary embodiment of the high address extension and high count extension logic.

FIG. 6 is a block diagram of a scatter/gather unit constructed in accordance with an embodiment of the present invention.

FIG. 7 is a timing diagram of an exemplary scatter/gather program cycle according to the present invention.

DESCRIPTION OF THE ILLUSTRATIVE EMBODIMENTS

With reference now to the Figures and in particular with reference to FIG. 1, a conventional computer, or PC, designated 10, is of the environment to which the invention has particular utility. Computer 10 which preferably, but not necessarily, is of a type utilizing an IBM personal computer or a similar system, includes a console housing 12 in which a circuit board containing the necessary circuitry including a microprocessor and BIOS chips, controllers, random access memory and other hardware are arranged. The computer will also include a video display 14 and a keyboard 16 connected to the housing 12 through cable 18. Mass storage media includes a hard disk drive within the housing and is non-accessible to the user, and user-accessible floppy disks, as well as, optionally, CD-ROM drives 20 and 22.

FIG. 2 is a block diagram of a computer system constructed in accordance with an embodiment of the present invention. The system includes a PCI bus 30, an ISA bus 32, with a plurality of ISA masters 36 and ISA slaves 38. A plurality of PCI memory slaves 40 are coupled to the PCI bus 30.

The bridge chip 34 contains an ISA interface 42 coupled between the ISA bus 32 and a system bus 44. A PCI interface 46 is provided between the PCI bus 30 and system bus 44. The bridge chip 34 also has a DMA controller 50, programmable I/O (PIO) registers 52, and a scatter/gather unit 54. The DMA controller 50 is coupled to the ISA bus 32. The bridge chip 34 provides an interface between the PCI bus 30 and the ISA bus 32.

The ISA bus interface 42 in the bridge chip 34 translates ISA bus cycles into system bus cycles for use by the bridge chip 34. The PCI bus interface 46 converts PCI bus cycles from the PCI bus 30 into system bus cycles for the bridge chip 34. The DMA controller 50 controls DMA control of memory accesses within the system. The DMA controller 50 provides a plurality of separate DMA channels over which memory accesses involving the individual ISA masters 36 are respectively communicated.

The DMA controller 50 is programmable over either the ISA bus 32 as in the prior art, or by the scatter/gather unit 54. Allowing the DMA controller 50 to still be programmed over the ISA bus 32 permits compatible PIO operation to occur when the DMA controller is programmed, so that if compatibility software is running, it appears that the DMA controller 50 is residing on the ISA bus 32. The operation of the scatter/gather unit 54 will be described later with respect to FIGS. 6 and 7.

Either the DMA controller 50 or the ISA bus master 36 can generate transfer cycles, since the DMA controller 50 acts as a bus master on the ISA bus 32. Both the ISA master 36 and the DMA controller 50 can access memory located on either the ISA bus 32 or the PCI bus 30. For ease of explanation in the following description, however, examples will be described in which the ISA bus master 36 is generating transfer cycles. When this occurs, the DMA controller 50 acts as an arbitration device.

When the ISA bus master 36 generates a transfer cycle, the ISA bus master 36 produces a memory address. This is the starting address and indicates the designation of transfer data.

As stated earlier, the 32-bit addressing capability of the PCI bus 30 provides a memory addressing capacity of 4 gigabytes. By contrast, since an ISA bus master 36 can only generate 24-bit addresses, it is limited to addressing memory within a 16 megabyte segment. The prior art solution uses external circuitry to set the most significant bit of a 32-bit PCI address high, when a master signal is detected on the ISA bus 32. This reallocates the 16 megabytes of memory to a specific different location within the 4 gigabytes of memory. However, all accesses to the memory on the PCI bus 30 by the ISA bus masters 36 would be to this same location. By contrast, the present invention provides memory access to any memory location within the 4 gigabytes of memory addressable over the PCI bus 30.

FIG. 3 is a block diagram of an exemplary embodiment of a DMA controller circuit 50 which provides dynamic configuration of the DMA channels. The DMA controller circuit 50 includes a pair of cascaded DMA controllers 60, 62 and a multiplexer circuit 64.

Preferred embodiments of the present invention use a commercially available DMA controller, such as the 8237 DMA controller manufactured by Intel. Each 8237 DMA controller provides four separate DMA channels which can be used independently for memory transfers. Certain well-known computer systems, such as the IBM PC/AT design, contain two 8237 DMA controllers. The DMA controller circuit 50 of FIG. 3 also uses two conventional DMA controllers 60, 62. One channel of the first DMA controller 60 is used to cascade the second DMA controller 62, as is known in the prior art. The pair of DMA controllers 60, 62 therefore provide a total of seven DMA channels, with four channels provided by the first DMA controller 60, and three channels provided by the second DMA controller 62.

A number of different signals, such as the clock signal, etc., have not been shown in FIG. 3 so as not to obscure the invention. However, one of ordinary skill in the art would recognize that these conventional signals would be used in practice.

The first and second DMA controllers 60, 62 produce sixteen bits (15:0) of the memory address to the PCI bus 30. As will be explained in more detail with respect to FIG. 5, the DMA controller circuit 50 also has a low page register, a high page register, and a high count register, although these registers can in certain embodiments be separate from the DMA controller circuit 50.

The first DMA controller 60 provides DMA channels 0–3 in response to request signals DREQ#0–DREQ#3. Addresses are programmed in the DMA controller 60 over lines ADR#0–ADR#3. Data is received over input lines DAT#0–DAT#7. Acknowledge signals are generated by the first DMA controller 60 as signals DACK#0–DACK#3.

The eight least significant bits of a 16-bit address are produced as output signals on pins A0–A7. The eight most significant bits of the 16-bit address are provided as outputs from D0–D7 when the address strobe signal ADSTB is enabled. When the address strobe is not enabled, the data signals are provided as the output from D0–D7. A hold request output (HRQ) from the first DMA controller 60 is coupled to the DREQ#0 input of the second DMA controller 62 and DACK#0 of the second DMA controller 62 is connected to the hold acknowledge input (hlda) of the first DMA controller 60 to provide the cascading of the pair of controllers 60, 62, as is well known in the art.

The second DMA controller 62 provides DMA channels 5–7 and receives DREQ#5–DREQ#7 at inputs DREQ#1–DREQ#3. The second DMA controller 62 pro-

vides at output pins DACK#1–DACK#3 the output signals DACK#5–DACK#7. The outputs of the address lines A0–A7 and data lines D0–D7 are the same as for the first DMA controller 60. Data lines D0–D7 receive as inputs DATA#0–DATA#7, the same as the first DMA controller 60. However, the second DMA controller 62 receives at inputs A0–A3 the signals ADR#1–ADR#4. This ensures that the first and second DMA controllers 60, 62 do not respond to the same I/O programming address.

The addresses generated by the first and second DMA controllers 60, 62 are received by the multiplexer circuit 64, shown in dashed lines. The multiplexer circuit 64 comprises two 8-bit address latches 66, 68, an 8-bit 2-to-1 multiplexer 70, and two 8-bit 4-to-1 multiplexers 72, 74.

The 8-bit 2-to-1 multiplexer 70 is a data multiplexer that receives the data signals D0–D7 from each of the first and second DMA controllers 60, 62. The data multiplexer 70 selects which set of data signals will be produced as output data (7-0) when reading the registers out of the first and second DMA controllers 60, 62.

When the data lines D0–D7 of the first and second DMA controllers 60, 62 are used as address lines for the most significant bits 15-8, these bits are latched in the respective 8-bit address latches 66, 68. The latches 66, 68 are under the control of the address strobe signal ADSTB from the first and second DMA controllers 60, 62.

The 8-bit 4-to-1 multiplexer 72 is a least significant bit multiplexer that produces as an output the least significant 8 bits of each address. The 8-bit 4-to-1 multiplexer 74 is a most significant bit multiplexer and produces the most significant bits 15-8 of each address.

The first multiplexer 72 has four inputs. At the first input are the least significant 8 bits of the address generated by the first DMA controller 60. At the second input, the first multiplexer 72 receives the least significant seven bits (6-0) of the least significant bits of the address generated by the first DMA controller 60. However, these bits are shifted to the left by one position and a zero is provided as the least significant bit of the 8-bits input into the multiplexer 72. The second input therefore represents the least significant bits of the memory address shifted by one position to the left, with a zero being placed into the least significant bit position. The third input of the first multiplexer 72 are the least significant 8 bits (7-0) of the unshifted memory address generated by the second DMA controller 62. The fourth input of the first multiplexer 72 receives a shifted memory address from the second DMA controller 62. This shifted memory address has been shifted in the same manner as the memory address received at the second input.

The second multiplexer 74 receives at its first input the most significant bits (15-8) of the address generated by the first DMA controller 60. These bits are provided by the 8-bit address latch 66. At its third input, the second multiplexer 74 receives the most significant bits (15-8) generated by addresses from the second DMA controller 62, through the 8-bit address latch 68. The second multiplexer 74 receives at its second and fourth inputs a shifted version of the most significant bits from the first and second DMA controllers 60, 62, respectively. In bit position 8, the output of address line A7 is provided. The upper seven bits in positions 15-8 are provided by bits (14-9) of the shifted memory address. Thus, the address bits provided at the second and fourth inputs correspond to bits (14-7) of the unshifted memory address, but shifted into positions 15-8. The second multiplexer 74 therefore receives the eight most significant bits of an unshifted memory address from each of the first and

second DMA controllers **60, 62** and the eight most significant bits of a shifted memory address from each of the first and second DMA controllers **60, 62**.

For reasons to be described later, it is necessary to preserve the shifted bit **15** so that the information contained in this bit is not lost. The shifted bit **15** is provided at outputs (one for each DMA controller **60, 62**) as address bit **16** (DMA1A16, DMA2A16).

The DMA control circuit includes a programmable register **76** which contains information for each DMA channel as to whether that channel is an 8-bit channel or a 16-bit channel. The acknowledge signals (DACK#0–DACK#3; DACK#5–DACK#7) for the seven DMA channels are provided as inputs to a seven-to-two multiplexer **77** that produces an output signal that indicates which DMA controller **60** or **62** carries the DMA channel that will be used for the transfer. Based upon the information in the register **76** and the signal from the multiplexer **77**, simple encoding logic **79** produces a select signal that causes the multiplexers **72, 74** to provide the required output.

In preferred embodiments of the invention, upon system power-up, the register **76** defaults to the IBM PC/AT design of configuring DMA channels **0–3** as 8-bit channels and DMA channels **5–7** as 16-bit channels.

Two examples of operation will be described. Assume for the first example that DMA channel **5** is currently programmed in the DMA control circuit to be an 8-bit channel. When a DMA control cycle is performed, only unshifted memory address should be produced as an output from the DMA control circuit. The signal from the encoding logic **79** corresponding to DMA channel **5** being an 8-bit channel causes the first multiplexer **72** to select the third input having the least significant bits (**7-0**) of the unshifted memory address in the second DMA controller **62** to be produced at the multiplexer output as address (**7-0**). The same control signal causes the second multiplexer **74** to select the most significant 8-bits of the unshifted memory address from the second DMA controller **62** to produce at the multiplexer output the address bits (**15-8**). Since these bits have not been shifted from the address generated by the second DMA controller **62**, either odd or even accesses to memory are possible, thereby providing 8-bit memory accesses.

In the second example, the DMA channel **2** is programmed to be a 16-bit channel provided by the first DMA controller **60**. The control signal from the encoding logic **79** causes the multiplexer **72** to select the second input which corresponds to the least significant bits of the shifted memory address and a zero in the least significant position. This is provided at the multiplexer output as address bits (**7-0**) of a memory access. The control signal also causes the second multiplexer **74** to select the most significant bits of the shifted memory address (second input) to be produced as the output address bits (**15-8**).

The above are examples only, as DMA channel **5** can be readily programmed to be a 16-bit channel, while DMA channel **2** can be programmed to be an 8-bit channel.

The circuitry of FIG. **3** provides the lower 16 bits of a memory address. The upper 16 bits (**31:16**) are produced by high address and high count extension logic **80**, as shown in the basic block diagram of FIG. **4**. The logic **80** also produces a count extension by providing the upper 8 bits (**23:16**) of a 24-bit transfer count, which in the prior art has been limited to 16 bits. The logic **80** can be considered as part of the DMA controller circuit **50** or a separate unit.

FIG. **5** is a block diagram of an exemplary embodiment of the high address and high count extension logic **80**. The

logic **80** includes a low page register **82**, a high page register **84**, and a high count register **86**. The low page register **82** has been used in prior art designs (such as the IBM PC/AT) to provide another eight (8) bits of memory addressing capability, for a total of twenty-four (24) bit addressing capability, or 16 megabytes. The high page register **84** of the present invention provides an additional eight bits of memory addressing capability for a total of thirty-two (32) bit addressing capability, or 4 gigabytes. These additional 8 bits are sent to the PCI bus **30** (concatenated with the lower 24 bits of the memory address) by the high page register **84** when an ISA bus master **36** gains control of a DMA channel. The contents of the high page register **84** can be loaded with different values for each of the seven different DMA channels, so that the seven ISA bus masters **36** are able to access seven different 16 megabyte segments of memory within the 4 gigabytes of memory on the PCI bus **30**. The high page register **84** is programmable, so that the allocation of a specific memory location for a 16 megabyte segment for an individual ISA bus master **36** can be changed.

The addition of a high page register to the existing design of a DMA controller circuit used in the IBM PC/AT computer system extends DMA transfer addresses from 24 bits to 32 bits, and all these bits may increment as well. This will be described below, as well as the features which allow incrementing and decrementing across any boundary. This was not possible in the original IBM PC/AT design, although it uses a low page register. Before describing the details of the logic **80**, the concept of boundary crossing will be explained.

Assume that the starting address of a DMA transfer is **1**. When a DMA controller is in an "increment" mode, the next DMA transfer will access address **2**. Similarly, a DMA controller can decrement the address on subsequent transfers. However, in prior art devices such as the IBM PC/AT, the address could not be incremented or decremented across "boundaries". A boundary is located between bytes of an address, for example, such as between bits **15** and **16**, and bits **23** and **24**. When in the increment mode the next address causes the bits below a boundary (for example, bits **15:0**) to become all zeros, and a 1 to enter the bit above the boundary (bit **16**), the boundary has been crossed. In the IBM PC/AT design, the lower 16 bits can increment across boundaries, but the next 8 bits (**23:16**) cannot do so. Because of this limitation, the maximum number of transfers that can be performed in a PC/AT compatible system is 64K bytes for each transfer request. After 64K bytes have been transferred, if the starting address was originally zero, the same memory location would be re-accessed by the next address.

Now referring to FIG. **5**, the CPU **41** initially programs the registers of a Programmed I/O control and registers unit **88** (hereinafter "PIO **88**") with the starting address, the transfer count and the mode. This information in the registers of the PIO **88** provides the DMA controller circuit **50** with the information it needs to perform a transfer. The PIO **88** and the high address extension architecture basically decode the address and I/O read and write commands to detect when to read or write several of the internal registers inside the first and second DMA controllers **60, 62** and the external registers for the DMA controller circuit **50**. The PIO **88** also generates chip select signals to the first and second DMA controllers **60, 62** for address decodes.

The chip select **1** signal is received at the chip select input of the first DMA controller **60**. The chip select **2** signal is received at the second DMA controller **62**. The chip select signals are asserted for cycles accessing the registers in the respective DMA controller **60, 62**. Inside the PIO **88** are

extended mode registers which are extensions to the mode registers inside the DMA controllers **60, 62**. The extended mode register is an 8-bit register. Five of the bits contain information which indicates to the DMA controllers **60, 62** whether the channel is an 8- or 16-bit channel (see register **76** in FIG. **3**), and whether the transfer will be a compatible cycle or a faster version. The PIO **88** receives the acknowledge DACK signals which are used to select the values to read out of the high and low page registers during DMA transfers and ensure that the correct channel information is routed to the address bus.

The low page register **82** contains 14 bytes of information. It contains the base address register for all seven channels, i.e. 7 bytes, and the current address register for all seven channels, another 7 bytes. When the low page register **82** is programmed, both the base and the current address registers are written simultaneously. As the DMA controller circuit **50** performs its transfer, the current address register in the low page register **82** is updated at the end of each writeback cycle.

The base address remains the initial value that was programmed in the base address register. This value supports a DMA mode termed "auto-initialize" in which upon completion of a DMA transfer, the base contents are restored to the current register and the same transfer can then proceed once again.

The high page register **84** is constructed essentially the same as the low page register **82**. The high page register **84** contains the high page information for base and current addresses and an associated multiplexer to steer the data out.

An 8/7 bit up/down counter **90** is the counter for address bits **23-16** and is loaded by the low page register **82** upon initiation of a DMA transfer and drives address bits (**23-16**) out onto the bus. Bit **16** passes through a multiplexer **92** so that the only bit that is actually multiplexed is bit **16**. In this mode, the counter **90** counts in a manner that accounts for the fact that the least significant bit of the address is now being driven up from the lower 16 bits by shifting.

In order to provide IBM PC/AT compatibility, in unshifted operation the 8/7 bit up/down counter **90** operates as a conventional 8 bit counter, since the least significant bit of the address is not being driven up from the lower 16 bits. When the low page register **82** is accessed, the high page register **84** is cleared (all zeros placed into the register **84**) and counting of the high address extension is disabled. The accessing of the lower order addresses or counts places "ff" into the high count register **86**. These actions cause the DMA control circuit **50** to operate and appear to the other devices as a PC/AT compatible DMA control circuit. The accessing of the high page register **84** automatically changes the DMA control circuit **50** from the PC/AT compatible mode to the high address extension mode in which high address counting is enabled..

A high page 8-bit up/down counter **94** is loaded with the contents of the high page register **84** upon initiation of the DMA transfer. Both the 8-bit up/down counter **94** and the 8/7-bit up/down counter **90** are cascaded with a carry-out bit. The 8/7 up/down counter carry-out bit feeds into the carry-in of the 8-bit up/down counter **94** to propagate counts across address boundaries. The carry-in to the 8/7-bit up/down counter **90** is driven by a 4-to-1 multiplexer **96** of boundary crossing detection logic **98**.

The up/down counters **90, 94** can also be referred to as "address counters".

The 4-to-1 multiplexer **96** selects its output in dependence on whether the DMA controller circuit **50** is in 8-bit mode

or 16-bit mode and whether it is incrementing or decrementing. The signals entering the AND gates **100-106** driving the multiplexer **96** are the address bits from the first and second DMA controllers **60** and **62**, i.e., the lower 16 bits of the address.

When the DMA is in an increment count by bytes mode, i.e. an increment by 8-bit mode, the first AND gate **100** of the multiplexer **96** is sampled. If "ff" (hexadecimal) is sampled, this indicates to the DMA controller circuit **50** that the address is about to cross a boundary upon a completion of the current transfer. If the current transfer is to "ffff", the next transfer should then be to 10000. To increment that bit properly, the carry-in bit will now become activated to the 8/7-bit up/down counter **90**.

If the DMA controller circuit **50** is presently in the 16-bit mode and an increment mode, the second AND gate **102** is sampled. A boundary will be crossed if the current address is "fffe" because the next address will be 10000. This situation also causes the carry-in to be provided to the 8/7-bit up/down counter **90**.

The third input to the multiplexer **90** from the AND gate **104** is sampled when the DMA controller circuit **50** is in the decrement mode and an 8-bit transfer mode (i.e., counting by bytes). When the address is 0000, the next decrement will cross a boundary. A carry is generated to the counter **90** to allow it to decrement properly so that the next count will be "ffff". This causes the 8/7-bit up/down counter **90** to decrement.

The fourth input to the multiplexer **96** from the AND gate **106** is used in the decrement mode and a 16-bit transfer mode. When the lower address ends in '010'b, this indicates to the 8/7-bit up/down counter **90** that there will be a decrement across the 16-bit boundary. Again, a carry is generated to the counter **90** so that the counter **90** will decrement properly across the boundary.

A 2-to-18-bit multiplexer **108** selects the contents of the low page register **82** or the contents of the high page register **84**. This multiplexer **108** allows programmed I/O reads of these registers **82, 84** and is decoded by the PIO **88**.

Preceding each of the low page and the high page registers **82, 84** is an 8-bit 2-to-1 multiplexer **110, 112**. The 8-bit 2-to-1 multiplexers **110, 112** selects either: the data from the internal system bus of the bridge chip **34** to the DMA controller circuit **50** to write the page registers **82, 84** initially; or the outputs of the up/down counters **90, 94** to do a write back operation to the page registers **82, 84**. This allows the current address which has just been incremented to be written back into the page registers **82, 84**.

To control this high address extension and high count extension logic **80**, a counter control state machine **114** is provided. The state machine **114** loads the contents of the low page and high page registers **82, 84** into the counters **90, 94** when either of the first or second DMA controllers **60, 62** becomes active. The state machine **114** increments those counters **90, 94** upon completion of a transfer cycle. As long as transfer cycles are occurring and the addresses are incrementing, count instructions will be allowed to change the counters if a carry-in is detected from the boundary crossing detection logic **98**. If a carry-in is not detected, the counters **90, 94** will not change their state, and so will not change their value. If an address boundary is crossed, the carry-in will be enabled and a count enable will be enabled by the counter control state machine **114** simultaneously to cause the counters **90, 94** to increment or decrement.

Upon completion of a DMA transfer cycle or series of DMA transfer cycles, the contents of the counters **90, 94** are

written back into the page registers **82, 84** by the counter control state machine **114**. The 8-bit 2-to-1 multiplexers **110, 112** at the inputs of the low page and high page registers **82, 84** are selected by the state machine **114** to use the counter output contents and perform a write back cycle into the low and high page registers **82, 84**. This completes the DMA transfer cycle and the current address register is updated.

The 2-to-1 multiplexer **116** receives bit **16** from the first and second DMA controllers **60, 62** as DMA1A16 and DMA2A16. The multiplexer is used to select between these shifted address 16th bits generated from the first or second DMA controller **60** or **62** for purposes of a 16-bit shifted address operation. The output of the multiplexer **116** is provided to the 2-to-1 multiplexer **92** which ultimately is selected to generate bit **16** of the address which is driven externally by the DMA controller circuit **50**. This allows the overwriting of bit **16** for 16-bit address shifter compatibility.

The high count register **86** provides a count extension to 24 bits. The operation of this high count register **86** is analogous to the address extension logic in that the same control signals from the counter control state machine **114** control the loading of the high count register **86** and the incrementing of the high count and writing back the high count value to the high count register **86**. The high count register **86** contains 7 bytes; only a current count for each of the seven DMA channels is stored.

As with the low and high page registers **82, 84**, the high count register **86** is coupled to an 8-bit 2-to-1 multiplexer **118** and an 8-bit down counter **120**. A down counter is used since the count is only decremented, and is not incremented.

A signal from the first and second DMA controllers **60** and **62** to an AND gate **122** detects when all internal bits from these controllers **60, 62** are 1 in the 16-bit transfer count. This is the indication to the high counter **120** to decrement.

The count signals are sampled directly to determine when a decrement boundary is going to be crossed in order to clock the high count extension. Unlike the address extension logic which is programmed through the PIO **88**, in preferred embodiments, the count extension is only programmable by a scatter gather program. Scatter gather programs are well-known to those of ordinary skill in the art.

With the present invention as described above, an ISA bus master **36** is able to operate in a conventional manner to address 16 megabytes. The accessing of memory above the 16 megabyte limit and the count extension is transparent to the ISA bus masters **36**, as it is performed by the bridge chip **34** using the DMA controller circuit **50**.

In order to provide backward compatibility, the DMA controller circuit defaults to PC/AT compatible mode, i.e. 16-bit addressing with a low page register. Once the high order byte is programmed, however the DMA controller circuit **50** changes from the default PC/AT mode into a 32-bit mode.

FIG. **6** is a block diagram of an exemplary embodiment of the scatter/gather unit **54** depicted in FIG. **2**. The scatter/gather unit **54** is connected to the DMA control circuit **50** on the bridge chip **34**.

The scatter/gather unit **54** has a PCI bus master **130**, a PCI slave **132**, a program controller **134**, and a multiplexer **136**. The PCI master **130** fetches descriptor control blocks from memory, which can be located on either the ISA bus **32** or the PCI bus **30**. The PCI slave **132** is coupled to the PCI bus **30** and functions as the programmed I/O registers for the scatter/gather unit **54**. The PCI slave **132** contains a plurality of registers which stores for the scatter/gather unit **54** the starting address of the descriptor table in memory, as well as status and commands.

The program controller **134** uses the data fetched from memory by the PCI master **130** and programs this data into the DMA control circuit **50**. This data includes a 32-bit DMA address, a 24-bit transfer count, and an end of link bit.

The multiplexer **136** receives from the program controller **134** at one set of inputs an I/O write signal (sgb_iow_n), an 8-bit data signal (sgb_data), and a 12-bit address signal (sgb_addr). The multiplexer **136** receives at another set of inputs that are coupled to the ISA bus **32** an I/O write signal (isa_iow_n), an 8-bit data signal (isa_data), and a 12-bit address signal (isa_addr). The address signal is an I/O address that selects the appropriate DMA register in which the data contained in the data signal will be written.

In response to a select signal (sgb_active) from the program controller **134**, the multiplexer **136** selects which of the sets of input signals are produced as output signals to program the DMA control circuit **50**. When the scatter/gather unit **54** is to program the DMA control circuit **50**, the select signal causes the multiplexer **136** to select the signals directly from the program controller **134**. When compatibility software is run on the system, the scatter/gather unit **54** is inactive and the select signal causes the multiplexer **136** to program the DMA control circuit **50** with the signals from the ISA bus **32**. The DMA control circuit **50** is not aware of the source of the programming signals.

The present invention takes advantage of the location of the scatter/gather unit **54** on the bridge chip **34** by performing the programming operations of the DMA control circuit **50** (also on the bridge chip **34**) at the 33 MHz speed of the bridge chip **34**. The scatter/gather unit **54** does not run any ISA cycles at 8 MHz, and is free from compatibility requirements of an ISA cycle. Although the present invention allows the DMA control circuit **50** to be programmed with ISA bus cycles for compatibility reasons, programming of the DMA control circuit **50** by the program controller **134** is approximately an order of magnitude faster than with ISA bus cycles.

An exemplary program cycle using the scatter/gather unit of the present invention is depicted in the timing diagram of FIG. **7**. In this example, the PCI master **130** has already fetched the descriptor information from memory. The scatter/gather unit **54** has been initiated by the CPU **41** and is therefore active. The select signal causes the multiplexer **136** to select the signals from the program controller **134** to directly program the DMA control circuit **50**, without running ISA bus cycles on the ISA bus **32**.

During clocks **1** and **2**, a byte pointer is cleared by the assertion of the address for a DMA byte pointer clear operation. A byte pointer is needed in the 8237 DMA controllers **60, 62** since only an 8-bit I/O address is used to program the 16-bit address in the DMA controllers **60, 62**. The byte pointer points to either the low order byte or the high order byte. The byte pointer is cleared in the first operation so that the pointer is pointing to the least significant byte so that the least significant byte in the DMA controller **60** or **62** will be written with the current address.

During clocks **1** and **2**, the data (chn1) sent to the DMA control circuit **50** is the DMA channel number (0-3; 5-7) that is being accessed. Note that the DMA control circuit **50** writes the address and data only when the write signal IOW# is active (low).

During clocks **3–6**, the current address registers within the DMA controllers **60, 62** are written to. The current_address is the address of the register within the DMA controllers **60, 62** that is being written to, and addr0 is the current address bits **7-0** for the DMA channel that is being accessed. The

data addr1 is the current address bits **15-8** for the DMA channel that is being accessed.

During clocks **7** and **8**, an address (lowpage) for the low page register **82** is provided and addr2, the current address bits (**23-16**) for the DMA channel, are written to this address. Similarly, the high page register **84** is written to in clocks **9** and **10**, with data addr3 (the current address bits (**31-24**) at the hi_page address.

The clear byte pointer operation and the writes to the current address registers are accesses directly to the internal design of the DMA controllers **60, 62**. These I/O writes are asynchronous because the DMA controllers **60, 62** latch the information on the rising edge of the IOW# signal itself. Thus, the address and data must remain valid for one clock after the actual I/O cycle to ensure that the hold times of the DMA controllers **60, 62** are met. By contrast, the low page register **82** and the high page register **84** are synchronous registers. The address and data need not remain valid for one clock after the actual I/O cycle. However, the address and data are driven one clock prior to the I/O cycle to ensure that setup times are not violated. The combination of these requirements causes the two clock delay between writing to the current address register and the low page register.

The byte pointer for the count is cleared in clock **11-13**, restoring the internal byte pointer of the DMA controllers **60, 62** to the lowest byte of the count. During these cycles, the data (chn1) sent to the DMA control circuit **50** is the DMA channel number (**0-3; 5-7**) that is being accessed.

The current_count address is the address where the current count bits (**15-0**) will be written to, during clocks **14-17**. The data being written to this address are cnt0 (current count bits (**7-0**)) and cnt1 (current count bits (**15-8**)).

In clocks **18** and **19**, the high count register is addressed (highcnt) and written to with cnt2 (current count bits (**23-16**)). The programming I/O cycle is then complete and the DMA control circuit **50**, now programmed, performs the DMA transfer.

The current count registers are asynchronous so that the address and data must remain valid for one clock after the actual I/O cycle to ensure that the hold times of the DMA controllers **60** and **62** are met. The high count register is synchronous and the address and data need not remain valid for one clock after the actual I/O cycle. However, the address and data are driven one clock prior to the I/O cycle to ensure that setup times are not violated. There is therefore a two clock delay between writing to the current count register and the high count register.

The scatter/gather unit **54** supports conventional scatter/gather functions, such as chaining through an end of link bit.

With the present invention, the amount of time required to run I/O program cycles to the DMA control circuit **50** are significantly reduced in comparison to the prior art. The I/O program cycles are "hidden cycles" since they are not run on the ISA bus and are thus not observable off the bridge chip. Very fast DMA register accesses can therefore be performed. For comparison purposes, with the present invention, an I/O write cycle can be performed in one 33 MHz clock. By contrast, in the prior art, each I/O write cycle requires several (for example eight) 8 MHz clocks on the ISA bus.

Although the present invention has been described and illustrated in detail, it is clearly understood that the same is by way of illustration and example only and is not to be taken by way of limitation, the spirit and scope of the present invention being limited only by the terms of the appended claims.

What is claimed is:

1. A bridge interface between buses of a computer system having a first bus and a second bus, the bridge comprising:
   a direct memory access (DMA) control circuit programmable by programming signals to perform a DMA transfer; and
   a scatter/gather unit coupled between the first bus and the DMA control circuit, the scatter/gather unit selectively providing the programming signals to the DMA control circuit directly or causing the programming signals to be provided over the first bus.

2. The bridge of claim **1**, wherein the first bus is an industry standard architecture (ISA) bus.

3. The bridge of claim **1**, wherein the scatter/gather unit includes a program controller that generates the programming signals that are selectively provided directly to the DMA control circuit.

4. The bridge of claim **3**, wherein the scatter/gather unit includes a multiplexer that has a first set of inputs coupled to the program controller to receive the programming signals directly from the program controller, and a second set of inputs coupled to the first bus to receive the programming signals from the first bus, and a select input which receives select signal indicating which of the first or second set of inputs should be provided as output signals of the multiplexer.

5. The bridge of claim **4**, wherein the DMA control circuit directly receives the output signals of the multiplexer as the programming signals.

6. The bridge of claim **5**, wherein the programming signals include an input/output (I/O) write signal, an address signal, and a data signal.

7. The bridge of claim **1**, wherein the programming signals are written to the DMA control circuit directly from the scatter/gather unit in I/O write cycles, each I/O write cycle being performed in one 33 MHz clock; and wherein the programming signals are written to the DMA control circuit over the first bus in first bus I/O write cycles, each first bus I/O write cycle being performed in a plurality of 8 MHz clocks.

8. A computer system comprising:
   a first bus and a second bus; and
   a bridge coupled between the first and second buses, the bridge having: a direct memory access (DMA) control circuit programmable by programming signals to perform a DMA transfer; and a scatter/gather unit coupled between the first bus and the DMA control circuit, the scatter/gather unit selectively providing the programming signals to the DMA control circuit directly or causing the programming signals to be provided over the first bus.

9. The computer system of claim **8**, wherein the first bus is an industry standard architecture (ISA) bus.

10. The computer system of claim **8**, wherein the scatter/gather unit includes a program controller that generates the programming signals that are selectively provided directly to the DMA control circuit.

11. The computer system of claim **10**, wherein the scatter/gather unit includes a multiplexer that has a first set of inputs coupled to the program controller to receive the programming signals directly from the program controller, and a second set of inputs coupled to the first bus to receive the programming signals from the first bus, and a select input which receives select signal indicating which of the first or second set of inputs should be provided as output signals of the multiplexer.

12. The computer system of claim **11**, wherein the DMA control circuit directly receives the output signals of the multiplexer as the programming signals.

13. The computer system of claim **12**, wherein the programming signals include an input/output (I/O) write signal, an address signal, and a data signal.

14. The computer system of claim **8**, wherein the programming signals are written to the DMA control circuit directly from the scatter/gather unit in I/O write cycles, each I/O write cycle being performed in one 33 MHz clock; and

wherein the programming signals are written to the DMA control circuit over the first bus in first bus I/O write cycles, each first bus I/O write cycle being performed in a plurality of 8 MHz clocks.

* * * * *